Learning Depth-aware Deep Representations for Robotic Perception

Lorenzo Porzi[1,2], Samuel Rota Bulò[2], Adrian Penate-Sanchez[3], Elisa Ricci[1,2], Francesc Moreno-Noguer[4]

***Abstract*—Exploiting RGB-D data by means of Convolutional Neural Networks (CNNs) is at the core of a number of robotics applications, including object detection, scene semantic segmentation and grasping. Most existing approaches, however, exploit RGB-D data by simply considering depth as an additional input channel for the network. In this paper we show that the performance of deep architectures can be boosted by introducing DaConv, a novel, general-purpose CNN block which exploits depth to learn scale-aware feature representations. We demonstrate the benefits of DaConv on a variety of robotics oriented tasks, involving affordance detection, object coordinate regression and contour detection in RGB-D images. In each of these experiments we show the potential of the proposed block and how it can be readily integrated into existing CNN architectures.**

***Index Terms*—RGB-D Perception; Visual Learning**

I. Introduction

SINCE the introduction of Microsoft Kinect, RGB-D data has been used in robotics and computer vision to address a large variety of tasks, including visual odometry, 3D object pose estimation, people tracking and activity recognition. The success of depth sensors can be partially ascribed to the fact that they provide a low-cost solution to a fundamental problem in robotics, *i.e.* the recovery of scale.

In the last few years, deep learning techniques have attracted the attention of robotics researchers, as they generally guarantee improved performance over traditional learning-based approaches in a wide range of applications and heterogeneous types of data (*e.g.* images, audio, text). Deep models have been applied to a number of robotics tasks involving RGB inputs, *e.g.* monocular depth prediction [1], 3D scene layout understanding [2], change detection in large 3D maps [3] and camera relocalization [4], [5], [6].

The popularity of Convolutional Neural Networks (CNNs) has also encouraged researchers to investigate the adoption of deep models for dealing with RGB-D inputs. In particular, the idea of considering CNNs to learn features describing both RGB and depth data has been proved beneficial in object detection [7] and recognition [8], [9], semantic segmentation [10] and grasping [11]. Most previous works, however, exploit RGB-D data by considering depth or hand-crafted descriptors derived from depth (*i.e.* surface normals, HHA features [7]) as additional input channels for task-specific CNN architectures. In this way, the scale information provided by depth sensors is not explicitly used within the network model.

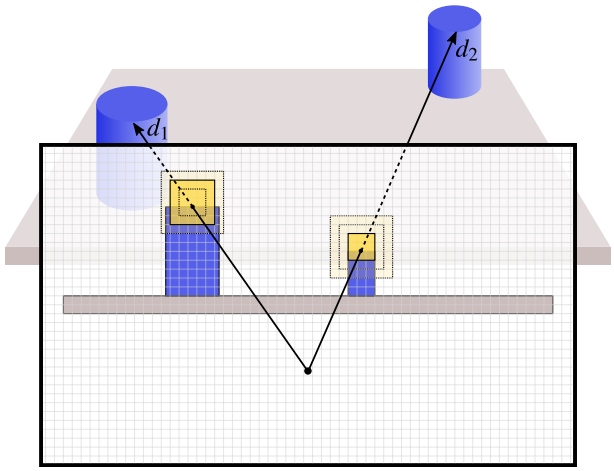

Fig. 1. Illustration of the intuition motivating our *Depth-aware Convolution* block. Two identical objects lying at different distances d_1, d_2 from the viewer appear to have different sizes on the image plane. It would be desirable, however, for them to activate the same convolutional neurons in a network. This can be achieved by locally tying the scale of the convolutional kernels to the measured depth.

In this paper we depart from previous works and we demonstrate that depth information can be directly used to derive more powerful CNN-based feature representations. Specifically, we introduce *DaConv* (*Depth-aware Convolution*), a novel, general-purpose block for CNN architectures which performs convolutions at multiple scales and combines the outputs using a learnable depth-dependent function. Intuitively, DaConv allows the network to learn convolutional activations that optimally adapt their receptive fields to the local scale of their input (see Figure 1).

In the experimental section we thoroughly demonstrate the benefits of DaConv in a number of robotics applications. In particular, we consider the tasks of affordance detection [12], 3D object coordinate regression [13], [14] and contour detection [15], [16]. In all three tasks we conduct experiments on publicly available benchmarks and show that our DaConv block can be used to systematically improve the performance

Manuscript received: September, 9, 2016; Accepted November, 17, 2016.

This paper was recommended for publication by Editor Jana Kosecka upon evaluation of the Associate Editor and Reviewers' comments. This work is partly funded by the Spanish MINECO project RobInstruct TIN2014-58178-R, by the ERA-Net Chistera project I-DRESS PCIN-2015-147, by the EU project AEROARMS H2020-ICT-2014-1-644271 and by the EU project SECOND HANDS H2020-ICT-2014-1-643950.

[1]Lorenzo Porzi and Elisa Ricci are with University of Perugia, Italy

[2]Lorenzo Porzi, Samuel Rota Bulò and Elisa Ricci are with Fondazione Bruno Kessler, Trento, Italy

[3]Adrian Penate-Sanchez is with University College London

[4]Francesc Moreno-Noguer is with Institut de Robòtica i Informàtica Industrial (UPC-CSIC), Barcelona, Spain

Digital Object Identifier (DOI): see top of this page.

of state-of-the-art CNN architectures. Specially remarkable are the improvements we obtain over state-of-the-art methods in the RGB-D Part Affordance dataset [12].

In short, the main contributions of this paper are twofold. First, we introduce DaConv, a novel depth-aware CNN computational block which uses depth information *within* the network to drive scale selection, departing from existing approaches that tackle invariance to generic transformations [17], [18] or scale [10], [19]. And second, we demonstrate that DaConv is a general-purpose block, which can be embedded in different CNN architectures, improving their performance. The code implementing DaConv will be made publicly available.

II. Related Work

CNNs for RGB-D data. Due to the impressive results achieved in tasks such as object recognition and detection, in the last few years CNNs have imposed themselves as the main learning paradigm in computer vision and robotic perception. Deep architectures have also been used to tackle challenging problems involving RGB-D data. For instance, Wu *et al.* adopted CNNs to address the problem of depth-based object recognition. Similarly, the tasks of multi-view recognition and next-best-view prediction are tackled in [20]. Here, Johns *et al.* developed a framework which considers a CNN model to classify image pairs and then optimally combines the obtained classification scores. Gupta *et al.* [21] used CNNs to learn how to align synthetic 3D models to real instances of the same object in RGB-D scenes, obtaining a significant improvement over previous works not considering deep models. Similarly, in the context of feature learning for RGB-D object recognition, Wang *et al.* [22] demonstrated that a CNN-based approach is advantageous over traditional learning based techniques. In [23] a CNN is trained to map image patches to a descriptor space where pose estimation and object recognition are solved using a simple nearest-neighbour technique. Deep models are considered in [11] to tackle the problem of grasping: CNNs are used to learn a grasp function and to compute a grasp quality score over all possible grasp poses, given a predefined discretization.

Learning Invariant Feature Representations with CNNs. A recent line of research on CNNs has addressed the problem of devising specific solutions for obtaining invariance to different kinds of transformations. Examples include the works of Bruna *et al.* [18], Gens *et al.* [24] and Laptev *et al.* [25], who sought invariance to translation and rotation, pose and part deformation, or generic transformations, respectively. Many works have focused on achieving invariance to scale changes. Common approaches include multi-scale pooling [26] and combining activations obtained from scaled versions of the input, either by simple concatenation [19] or by linear combination [27]. The method of Chen *et al.* [27] in particular has some similarities to ours: feature maps computed at different scales are linearly combined using an attention-like mechanism [28]. Differently from our approach, however, they do not exploit depth information as a prior and only apply their scale-aware mechanism at a single level in the network. Recently, in [29] a multi-scale convolutional network architecture is proposed to jointly perform depth prediction, surface normal estimation, and semantic labeling. An overview of approaches modeling scale changes within deep networks is provided in [16]. Differently from all these previous works, our approach uses depth information to drive the scale selection of the convolutional filters.

Specific efforts to define a common framework for CNN architectures focusing on learning invariant representations has been made in [17] and [30]. The former work presented the *Spatial Transformer* layer, which automatically learns a spatial transformation of its input. The work in [30] introduced an adaptation method to compute convolutional kernels. Similar to our DaConv, a local adaptation strategy is considered, motivated by the fact that different image regions may demand different adaptation functions. However, their tree-structured kernel adaptive CNN greatly differs from our DaConv block: since the focus of [30] is on facial traits recognition, kernels are dynamically updated according to the spatial distribution of facial landmarks rather than depth.

Traditionally, depth information has been used to achieve invariance to scale changes, *e.g.* in conjunction with random forests [31], [13]. In these methods, depth is used to determine the scale at which the binary features of a decision forest are calculated. More recently, some attempts have been made to derive deep models robust to scale using depth information: in [10], a global depth-dependent scaling is applied to the input of a CNN to solve a semantic segmentation task. However, in [10] the mapping between depth and scale is predefined, while in our approach the network *learns* how to use depth to *locally* handle scale at the convolutional filter level.

III. Learning Depth-Aware Convolutions

In this section we describe the key component of our contribution, namely a computational block for CNNs called *DaConv*, which can be regarded as a convolutional layer endowed with the ability to adapt the scale of the filter kernels based on depth information. One issue that we face is the impossibility of knowing a priori which pixels, and therefore which depths, contribute to activations within the DaConv block,[1] while we need this information to drive the scale selection. To sidestep this problem, we introduce an additional network (*DepthNet*) fed with depth information, working in parallel with the main network (*PredictionNet*). The role of DepthNet is to provide the DaConv blocks in PredictionNet with depth-related features that will trigger the decision about which scale to choose within each block. PredictionNet, instead, is devoted to delivering the final prediction. We call DaConvNet the entire architecture, which includes DepthNet and PredictionNet. In the remainder of this section we provide some more details about the proposed architecture. Additional details about the DaConvNet's inputs, outputs and training procedure are postponed to the experimental section.

Convolutions with scaled kernels. Within the DaConv block, we simulate convolutions with filter kernels at different scales via so-called *dilated* (*a.k.a. atrous*) convolutions [32]. A ℓ-dilated convolution is a standard convolution with a dilated

[1] We only have coarse information about theoretical receptive fields.

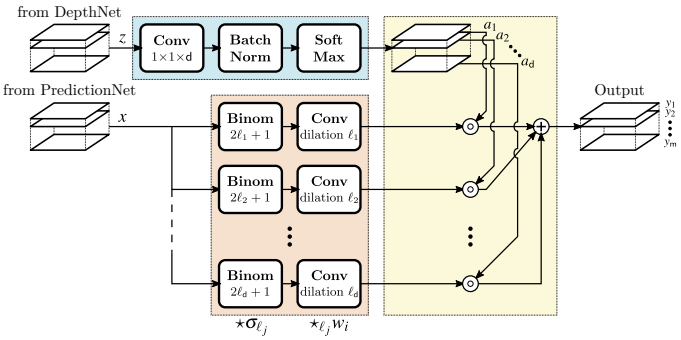

Fig. 2. Schematic representation of a DaConv block. Light blue block: computation of the scale selection factors a_j; orange block: convolution at different scales; yellow block: linear combination.

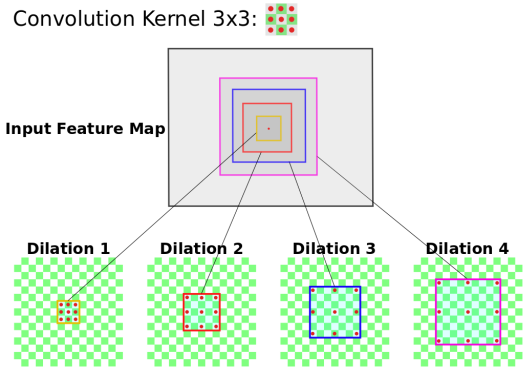

Fig. 3. In ℓ-dilated convolution the elements of a convolutional kernel are interspersed with $\ell-1$ zeroes, thus increasing the receptive field without adding extra parameters.

version of the filter, which is obtained by adding $\ell-1$ zeros between adjacent filter elements. More precisely, let $x, \omega : \mathbb{Z}^2 \to \mathbb{R}$ be a discrete function and a discrete filter kernel, respectively (we consider the 2D case for the sake of simplicity). The ℓ-dilated convolution of x and ω is given by

$$(x \star_\ell \omega)(\mathbf{r}) = \sum_{\mathbf{t}\in\mathbb{Z}^2} x(\mathbf{r}-\ell\mathbf{t})\omega(\mathbf{t}), \tag{1}$$

where $\mathbf{r} \in \mathbb{Z}^2$ and $\ell \in \mathbb{N}_{>0}$ is the *dilation factor*. One recovers the standard convolution by taking $\ell = 1$, *i.e.* $\star = \star_1$.

Before applying the ℓ-dilated convolution, we smooth the input signal x by convolving it with a smoothing kernel σ_ℓ in order to propagate local information, which would otherwise be lost due to the dilation operation. We implement σ_ℓ as a binomial kernel with window size $2\ell+1$, which ensures that we have stronger smoothing effects at higher scales, (or equivalently with larger dilation factors).

DaConv block.[2] This block extends standard convolutional layers with a data-driven selection of the filters' scale. It is fed with an input x computed by the previous layers of PredictionNet, and a depth-dependent input z from DepthNet (see Figure 2). Both x and z share the same spatial resolution, while they might have a different number of feature channels. Like a standard convolutional layer, DaConv is parametrized by m filter kernels $\{\omega_1, \ldots, \omega_\mathsf{m}\}$ (we omit the corresponding bias parameters), but in addition it has also d filter kernels $\{\nu_1, \ldots, \nu_\mathsf{d}\}$ with spatial resolution 1×1 that will be involved in the scale selection. Indeed, each filter ν_j is associated with a pre-fixed dilation factor ℓ_j. The output dimensionality of the block is the same one would expect from a standard convolution with the filter banks $\{\omega_1, \ldots, \omega_\mathsf{m}\}$.

We let the scale selection vary across different spatial locations. To do this, the input z from DepthNet is convolved with each filter ν_j and the output batch-normalized [33] before entering a softmax layer acting along the *feature* dimension. This operation preserves the input spatial resolution and yields a probability vector for each spatial location, indicating the scale selection distribution. The probability that dilation factor ℓ_j is chosen for spatial location $\mathbf{u}$ is denoted by $a_j(\mathbf{u})$ (see Figure 2 top). The use of batch-normalization before the softmax operation ensures that the scale selection will not be biased towards a fixed one across the entire dataset.

Finally, the scale selection probabilities encoded in $\{a_1, \ldots, a_\mathsf{d}\}$ are used by DaConv to linearly combine the outputs of convolutions of x with the filter kernels ω_i undergoing different dilation factors. In formal terms, the output of the DaConv block is given by

$$y_i = \sum_{j=1}^{\mathsf{d}} a_j \circ (x \star \sigma_{\ell_j} \star_{\ell_j} w_i), \tag{2}$$

where $\circ$ denotes the Hadamard (*a.k.a.* elementwise) product, $i \in \{1, \ldots, \mathsf{m}\}$ indexes one of the filter kernel ω_i, and we find between parentheses the smoothing operation and ℓ_j-dilated convolution previously described.

DepthNet. This network provides the depth-specific feature representations z that drive the selection of the dilation factors within each DaConv block in PredictionNet. It is designed in a way to ensure that the input z provided to each DaConv block has a spatial resolution that matches the one of the input x to the same DaConv block. Details about the actual topology of DepthNet in the different application scenarios are provided in the experimental section. Similarly, we postpone implementation details about PredictionNet.

IV. Depth-aware CNN Architectures for Robotic Perception Tasks

In this section we describe and evaluate the proposed depth-aware approach in three different tasks, which involve RGB-D data and are of interest for the robotics community, namely part affordance detection, object coordinates regression and contour detection. Each task requires pixel-level prediction models. Therefore, for each application, we consider as baseline a fully-convolutional network, as it currently represents the most common architectural choice for pixel-wise classification tasks. In order to demonstrate the advantages of our proposal, we systematically compare each baseline network with an associated DaConv network ("-DA" suffix in the tables and figures).

Each DaConv network is constructed by replacing some of the convolutional layers of the corresponding baseline netowrk with DaConv blocks, obtaining the PredictionNet, and pairing it with a similarly-structured DepthNet. Additional details

[2]We use the term "block" instead of "layer" because it can be built by composing standard layers found in recent deep network frameworks.

TABLE I
NETWORK ARCHITECTURES

Network		Architecture
Sec.IV-A	Baseline	$\texttt{c}[3,16];\texttt{c}[3,16];\texttt{p}[2,2] \rightarrow \texttt{c}[3,32];\texttt{c}[3,32];\texttt{p}[2,2] \rightarrow \texttt{c}[3,64];\texttt{c}[3,64];\texttt{p}[2,1] \rightarrow \texttt{c}[3,64];\texttt{c}[1,\texttt{n}]$
	DepthNet	$\texttt{c}[3,8];\texttt{c}[3,8];\texttt{p}[2,2] \rightarrow \texttt{c}[3,16];\texttt{c}[3,16];\texttt{p}[2,2] \rightarrow \texttt{c}[3,32];\texttt{c}[3,32]$
	PredictionNet	$\texttt{c}[3,16];\texttt{dac}[3,16];\texttt{p}[2,2] \rightarrow \texttt{c}[3,32];\texttt{dac}[3,32];\texttt{p}[2,2] \rightarrow \texttt{c}[3,64];\texttt{dac}[3,64];\texttt{p}[2,1] \rightarrow \texttt{c}[3,64];\texttt{c}[1,\texttt{n}]$
Sec.IV-B	Baseline	$\texttt{c}[7,64];\texttt{p}[2,2] \rightarrow \texttt{c}[7,64];\texttt{p}[2,2] \rightarrow \texttt{c}[7,64];\texttt{p}[2,2] \rightarrow \texttt{c}[7,64];\texttt{p}[2,2] \rightarrow$ $\texttt{u}[2,2];\texttt{d}[7,64] \rightarrow \texttt{u}[2,2];\texttt{d}[7,64] \rightarrow \texttt{u}[2,2];\texttt{d}[7,64] \rightarrow \texttt{u}[2,2];\texttt{d}[7,64] \rightarrow \texttt{c}[1,7]$
	DepthNet	$\texttt{c}[7,8];\texttt{p}[2,2] \rightarrow \texttt{c}[7,8];\texttt{p}[2,2] \rightarrow \texttt{c}[7,8];\texttt{p}[2,2] \rightarrow \texttt{c}[7,8]$
	PredictionNet	$\texttt{dac}[7,64];\texttt{p}[2,2] \rightarrow \texttt{dac}[7,64];\texttt{p}[2,2] \rightarrow \texttt{dac}[7,64];\texttt{p}[2,2] \rightarrow \texttt{dac}[7,64];\texttt{p}[2,2] \rightarrow$ $\texttt{u}[2,2];\texttt{d}[7,64] \rightarrow \texttt{u}[2,2];\texttt{d}[7,64] \rightarrow \texttt{u}[2,2];\texttt{d}[7,64] \rightarrow \texttt{u}[2,2];\texttt{d}[7,64] \rightarrow \texttt{c}[1,7]$
Sec.IV-C	Baseline	$\texttt{c}[3,64];\texttt{c}[3,64];\texttt{p}[2,2] \rightarrow \texttt{c}[3,128];\texttt{c}[3,128];\texttt{p}[2,2] \rightarrow \texttt{c}[3,256];\texttt{c}[3,256];\texttt{c}[3,256];\texttt{p}[2,2] \rightarrow$ $\texttt{c}[3,512];\texttt{c}[3,512];\texttt{c}[3,512];\texttt{p}[2,2] \rightarrow \texttt{c}[3,512];\texttt{c}[3,512];\texttt{c}[3,512]$
	DepthNet	$\texttt{c}[3,8];\texttt{c}[3,8];\texttt{p}[2,2] \rightarrow \texttt{c}[3,16];\texttt{c}[3,16];\texttt{p}[2,2] \rightarrow \texttt{c}[3,32];\texttt{c}[3,32];\texttt{c}[3,32]$
	PredictionNet	$\texttt{c}[3,64];\texttt{dac}[3,64];\texttt{p}[2,2] \rightarrow \texttt{c}[3,128];\texttt{dac}[3,128];\texttt{p}[2,2] \rightarrow \texttt{c}[3,256];\texttt{c}[3,256];\texttt{dac}[3,256];\texttt{p}[2,2] \rightarrow$ $\texttt{c}[3,512];\texttt{c}[3,512];\texttt{c}[3,512];\texttt{p}[2,2] \rightarrow \texttt{c}[3,512];\texttt{c}[3,512];\texttt{c}[3,512]$

Convolution: c[size, filters]; Deconvolution: d[size, filters]; DaConv: dac[size, filters]; Pooling: p[size, stride]; Unpooling: u[size, stride]

about the networks' architectures are provided in Table I, as well as in Sections IV-A, IV-B and IV-C.

Unless otherwise stated, we train all the networks using the Adam stochastic gradient descent method with a weight-decay factor of 5×10^{-5}, parameters $\beta_1 = 0.9, \beta_2 = 0.999$, and we use DaConv blocks with $\mathsf{d} = 3$ dilation factors, namely $\ell_j = 2^{j-1}$ for $i \in \{1,2,3\}$.[3] In the following sections we use the notation $f_i : \mathbb{Z}^2 \rightarrow \mathbb{R}$ to indicate the output of the ith channel of any network under consideration. All our networks are implemented using the Caffe [4] framework and trained on a single Nvidia K40 GPU.

A. Object coordinates regression

Several recent methods [13], [14] to estimate the pose of known 3D objects from a single RGB-D image share a common two-steps pipeline: (i) for each pixel in the image, predict its 3D coordinates in the object's frame of reference; (ii) geometrically estimate the object's pose from the correspondences between the predicted object coordinates and the observed depth. The first step results in a pixel-wise, vector-valued regression that is, in general, quite hard to solve. To simplify learning, previous work [13], [14] resort to a quantization of the object coordinates space, turning the problem into a classification one. In these works, the classification is performed using a random forest, while we show how the accuracy of step (i) can be improved by employing a DaConv network to perform the pixel-wise classification.

Dataset and experimental protocol. As in [13], [14], we consider the dataset of Hinterstoisser *et al.* [34], which comprises 15 sequences of RGB-D images of several objects lying on a cluttered table. For each sequence, we are given the 6-DOF pose of a specific object relative to the camera and a 3D mesh of the object. As in [13], we partition each dimension of an object's coordinates space into 5 uniform intervals, obtaining $5 \times 5 \times 5 = 125$ spatial bins in total. By doing so, we can rephrase the regression task into a classification task. The number of actual classes can be reduced, since only k (out of 125) spatial bins will contain at least one point from the object's surface, thus being a relevant coordinate for the pose estimation. Given the depth and camera pose information, we assign to each of the sequence's pixels a label in $\{1,\ldots,\mathsf{k}\}$ if it back-projects to one of the k relevant bins, or $\mathsf{k}+1$ if it belongs to the background.

In our experiments we randomly split each sequence into train, validation and test sets comprising, respectively, 30%, 10% and 60% of the images. All our results are obtained by training a different classifier on a train set, selecting parameters on the corresponding validation set and evaluating on the test set. Experimental results are reported in terms of the average per-class accuracy.

Network architecture and training. For this application we adopt a fully-convolutional network architecture reminiscent of the VGG net of Simonyan *et al.* [35]. Compared to [35], we drastically reduce the number of convolutional filters and exclude the fully-connected part of the network, as we want to obtain pixel-wise predictions. Furthermore, we feed the network with 6-channels tensors obtained by stacking the RGB image with the 3-channel surface normals computed from the depth. The architecture, summarized in Table I, is composed of four main blocks of 3×3 convolutions of stride 1, followed by 2×2 max pooling. The first two max pooling layers have stride 2, while the third one has stride 1, resulting in a final downsampling factor of 4. Because of this, the network outputs pixel-wise predictions at one fourth of the original resolution. At test time we up-sample the predictions using nearest-neighbor interpolation. As for DaConvNet, we replace the second convolutional layer within each of the first three blocks with a DaConv block.

The objective we use for training consists of a per-pixel softmax log-loss. The contribution of each pixel is opportunely weighted in order to compensate for the highly imbalanced class distribution in the dataset. In formal terms, we address the following optimization problem:

$$\arg\min - \sum_{\mathbf{r}} \xi_{l(\mathbf{r})} \log \frac{\exp(f_{l(\mathbf{r})}(\mathbf{r}))}{\sum_{i=1}^{\mathsf{k}+1} \exp(f_i(\mathbf{r}))}, \tag{3}$$

[3] In our experiments we found $\mathsf{d} = 3$ to be a good compromise between classification accuracy and computational complexity.

[4] http://caffe.berkeleyvision.org/

where $l(\mathbf{r}) \in \{1,\dots,\mathsf{k}+1\}$ is the ground truth label at spatial location $\mathbf{r} \in \mathbb{Z}^2$, and the minimization is implicitly taken with respect to the network parameters. The class-rebalancing weights ξ_i are defined as in [29], for all $1 \leq i \leq \mathsf{k}+1$:

$$\xi_i = \frac{\text{median}_i(\xi'_i)}{\xi'_i}, \tag{4}$$

$$\xi'_i = \frac{\#\text{pixels of class } i}{\#\text{pixels in images containing } i}. \tag{5}$$

As mentioned already at the beginning of this section, we optimize (3) via stochastic gradient descent. Both the baseline and DaConvNet are trained with the following schedule: 50 epochs with learning rate 10^{-2} followed by 25 epochs with learning rate 10^{-3}, batch size equal to 64. As is common practice when considering small datasets [16], we perform data augmentation during training. In particular, we form training batches by sampling a randomly rotated, scaled and translated 128×128 pixels patches from the training images. At test time we apply the learned network on full-resolution images.

Results. Figure 4 reports the results of our experimental evaluation, comparing the proposed DaConv architecture with the baseline, fully-convolutional network. The proposed network (CNN-DA) outperforms the baseline CNN using standard convolution layers for all different objects. On average, the use of DaConv blocks improves classification accuracy by 10%. As a reference, we also report the results obtained considering a random forest classifier, as this represents the common choice for coordinate regression tasks [13], [14]. In particular, we use the implementation in Piotr Dollár's toolbox [5], training a forest with 10 trees observing the same RGB plus surface normal inputs as the CNN. Note that this implementation differs from the one in [13], as the code for that is not publicly available. As one can see, deep architectures outperform off-the-shelf random forests, which is not so surprising, as CNNs currently achieve state-of-the-art results in many tasks in robotic perception. Figure 5 shows an example of the a_j functions learned by the first DaConv block of CNN-DA for the Ape object. Interestingly, the functions mostly follow the scene's depth, with gradually higher weights being assigned to the scale $j = 3$ in correspondence of closer objects, and vice-versa for $j = 1$. This is in accordance with the intuition illustrated in Figure 1.

B. Part Affordance Detection

The problem of localizing and identifying part affordances [12] is a fundamental task for deploying the next generation of robotic platforms, which are supposed to effectively collaborate with humans in everyday workspaces. Part affordance detection requires to segment and label image regions corresponding to object parts according to the interaction modality, or *affordance*. In other words, each affordance constitutes a class in the segmentation problem. Predicting affordances is very challenging since objects from different categories, with different shapes and visual appearances, can have parts associated to the same affordance.

Dataset and experimental protocol. In our experiments we consider the RGB-D Part Affordance Dataset of Myers *et al.* [12], covering 105 different tools and 7 different affordances, namely "grasp", "cut", "scoop", "contain", "pound", "support" and "wrap-grasp", for a total of about 30k images.¡ This dataset is split in two parts: (i) a Non-cluttered subset comprising RGB-D video sequences of single tools lying on a rotating plane; (ii) a Cluttered subset comprising 3 RGB-D video sequences of several different tools amassed over a table. One third of the video frames have been manually labeled by a group of users, and the labels automatically propagated to the remaining frames. To account for possibly discording labellings provided by different users, each pixel retains as ground-truth information a ranking of affordance labels, ordered from the most voted to the less voted one.

We follow the experimental protocol in [12] by directly using the publicly available evaluation code from the authors,[6] which considers only the manually labeled frames, both for training and testing, and gives separate results for the Non-cluttered and Cluttered subsets. Detection accuracy is evaluated in terms of three different metrics: *weighted F-measure* F^w_β, *rank weighted F-measure* R^w_β and *ranked correlation score* τ_k. For a detailed description of the way these metrics are calculated, we refer the reader to [12] and the public code.

Network architecture and training. We adopt the SegNet-Basic architecture in [36], summarized in Table I. This is a symmetric architecture that takes RGB images as input and is composed of four convolutional and four deconvolutional layers with 64 7×7 filters and stride 1. Each convolutional layer is followed by a 2×2, stride 2 max-pooling layer and each deconvolutional layer is preceded by a 2×2, stride 2 max-unpooling layer. Batch normalization is applied to the output of all convolutional and deconvolutional layers. The network output layer has 8 channels, corresponding to the 7 affordance classes with an additional background class. For our DaConvNet, we replace each convolutional layer with a DaConv block.

We train our networks by solving the following optimization problem:

$$\arg\min L_c + \lambda L_r\,, \tag{6}$$

where the minimization is intedend with respect to the network parameters. The objective is composed by a classification loss term L_c and a ranking-related loss term L_r. The classification loss L_c is a weighted sum of pixel-wise log-loss terms defined similarly to (3), where the per-pixel log-loss term is computed with respect to the top-ranked class in the ground-truth ranking. The ranking loss, L_r, is a sum of pixel-wise loss terms, each aimed at exploiting the ranking information from the ground-truth. It is defined as follows:

$$L_r = -\sum_{\mathbf{r}} \sum_{i \neq j} p_{i,j}(\mathbf{r}) \log(\sigma(f_i(\mathbf{r}) - f_j(\mathbf{r}))), \tag{7}$$

where $p_{i,j}(\mathbf{r})$ is 1 if the affordance i ranks higher than j in the ground-truth ranking for pixel $\mathbf{r}$, 0.5 if they have the same ranking and 0 otherwise, while $\sigma(\cdot)$ is the sigmoid function.

[5] https://pdollar.github.io/toolbox/

[6] http://www.umiacs.umd.edu/~amyers/part_affordance/

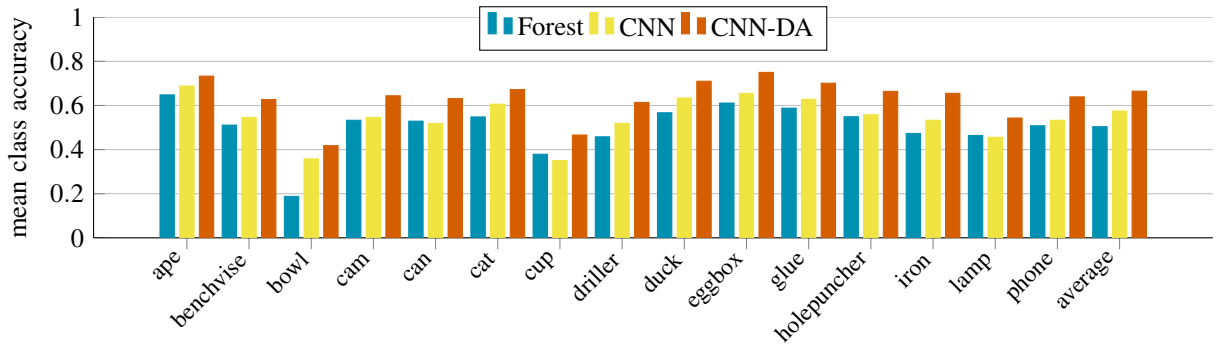

Fig. 4. Object coordinates regression results on the dataset of Hinterstoisser *et al.* [34].

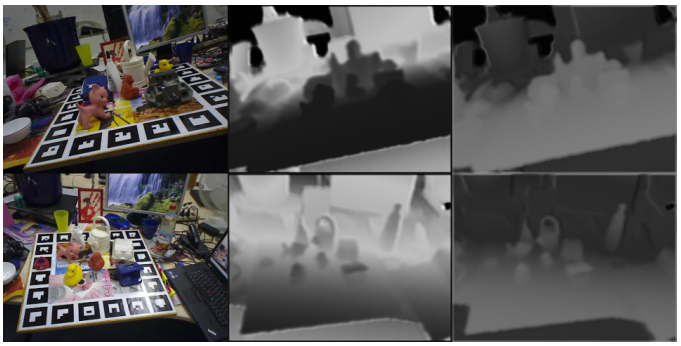

Fig. 5. Visualization of the functions a_1, a_3 (second and third column) learned by the network used in Section IV-A when applied to two frames from the coordinates regression dataset (first column). Note how a_1 (*i.e.* the weights of the finest scale) has higher values for distant objects, while a_3 (*i.e.* the weights of the coarsest scale) has higher values for close objects.

TABLE II
PART AFFORDANCE DETECTION RESULTS

Method	Non-cluttered			Cluttered		
	F_β^w	R_β^w	τ_k	F_β^w	R_β^w	τ_k
CNN	0.594	0.098	0.840	0.512	0.201	0.780
CNN-DA	**0.652**	**0.104**	**0.913**	**0.558**	**0.223**	**0.816**
S-HMP [12]	0.557	0.082	0.751	0.250	0.105	0.563
SRF [12]	0.460	0.081	0.643	0.165	0.083	0.435

Note that L_r can be seen as the sum of a set of cross-entropy losses, one for each possible (unordered) pair of affordances. From this point of view, $\sigma(f_i(\mathbf{r}) - f_j(\mathbf{r}))$ and $p_{i,j}(\mathbf{r})$ are, respectively, the predicted and ground-truth probabilities of i ranking higher than j for spatial location $\mathbf{r}$. The parameter λ (fixed to 0.125 in our experiments) weights the two terms of the objective.

Both the baseline network and DaConvNet are trained with the following schedule: 20k iterations with learning rate 10^{-3} followed by 10k iterations with learning rate 10^{-4}, batch size equal to 16. As in [12], we halve the resolution of the images for training and testing. Additionally, we normalize all input data by subtracting from each image the mean and dividing by the standard deviation of each channel (both computed on the training set).

Results. The results of our experimental evaluation are reported in Table II, where CNN and CNN-DA refer, respectively, to the baseline and DaConv versions of the CNNs. Our approach is compared to the state-of-the-art methods on this dataset, *i.e.* the Structured Random Forest and Sparse Coding methods presented in [12] (respectively SRF and S-HMP in the table). Differently from our CNN architectures, the classifiers in [12] do not take depth directly as input channel. Instead, a set of robust hand-crafted geometric features is derived from the depth channel, including surface normals and curvature features. From the reported results we observe that CNN-based models are more accurate than SRF and S-HMP in predicting affordance labels. Moreover, the best performances are achieved by the introduction of our DaConv block, both in the Clutter and in the Non-clutter datasets experiments. Focusing on each affordance category separately (Figure 6) we observe that, in the cluttered subset, CNN-DA produces the best results in terms of F_β^w for all categories, while in the non-cluttered subset it is outperformed by S-HMP and CNN in the "pound" and "support" categories, respectively. Moreover, some sample segmentation outputs for the three scenes of the Cluttered subset are shown in Figure 7.

We perform an additional set of experiments, where we modify our CNN by extending its RGB input with 5 additional channels, namely the 3-channels *surface normals* and the 2-channels *principal curvature* features, as defined in [12]. The addition of these hand-crafted features has a notable impact on the performance of the baseline network, where F_β^w increases from 0.594 to 0.642 and τ_k from 0.840 to 0.924 on the Non-cluttered subset. Conversely, in the DaConv network the performance increase is much more modest ($F_\beta^w = 0.667, \tau_k = 0.931$ in the Non-cluttered subset), suggesting that our approach is already able to capture most of the geometric information provided by the additional features.

C. Contour detection

Edge features are widely used in robotic perception for applications such as pose estimation, visual odometry and SLAM [37], [38] and are closely related to semantic segmentation tasks. Currently, learning-based techniques [15], [16] represent the state of the art for boundary detection. In this subsection we describe a CNN architecture based on DaConv for predicting contours in RGB images. Differently from the application scenarios described in the previous subsections, we consider a more challenging setting for testing the proposed

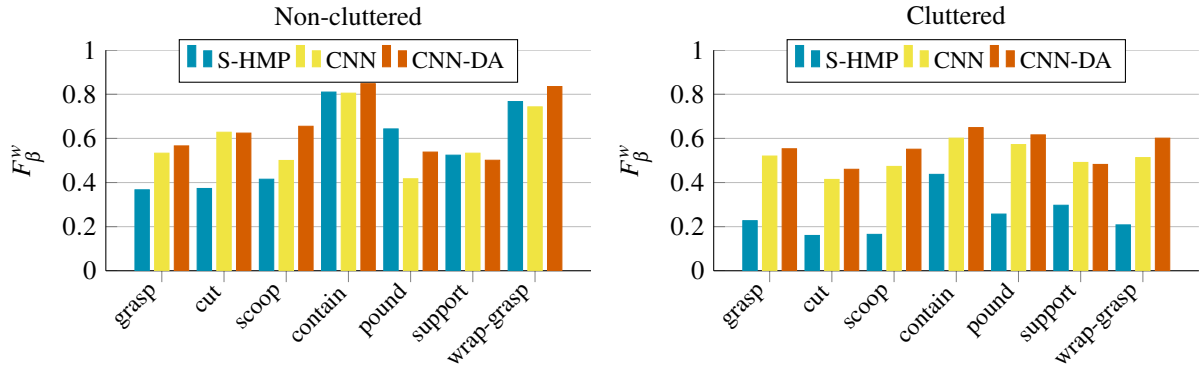

Fig. 6. Weighted F-measure on the Non-cluttered (left) and Cluttered (right) subsets of the RGB-D Part Affordance dataset, plotted by affordance.

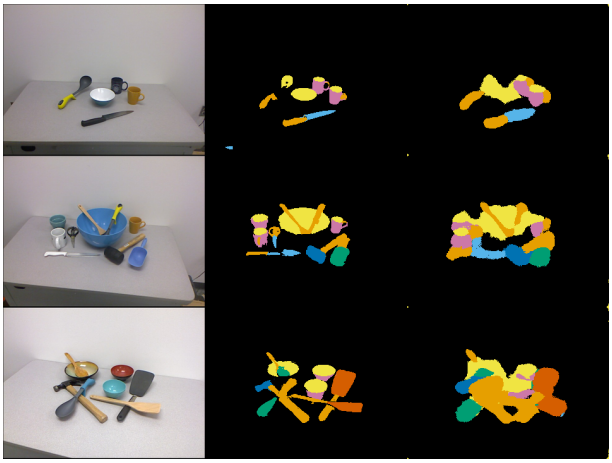

Fig. 7. Sample part affordance detection results. First column: input images. Second column: ground truth. Third column: prediction outputs obtained with the proposed DaConv architecture.

approach, *i.e.* we assume that depth images are not available but are "hallucinated" from RGB data. Specifically, motivated by recent research [29] demonstrating that the depth of a scene can be accurately estimated from a single RGB image, we adopt the depth prediction method in [29] to generate the depth maps used by DaConvNet. We show that in this way the performance of an existing RGB-only CNN for boundary detection can be improved without employing real depth data at test time.

Dataset and experimental protocol. We consider the NYUv2 dataset [39]. This dataset includes 407 024 RGB-D frames from 464 different indoor scenes, split into train and test subsets containing 249 and 215 scenes, respectively. A subset of 1449 frames, uniformly distributed among the scenes, has been densely annotated by assigning each pixel to one of several object classes and also by distinguishing between object instances. We follow the experimental protocol of [16], which uses 414 of the 795 annotated frames belonging to the train scenes for validation and leaves the remaining 381 for training. It also infers the object contour information from the semantic labeling. To evaluate contour detection performance we adopt the ODS, OIS and AP metrics, commonly used in previous works [16], [15], evaluating on the 654 annotated frames from the test scenes.

TABLE III
EDGE DETECTION RESULTS ON THE NYUV2 DATASET

Method	ODS	OIS	AP
gPb-ucm (RGB) [40]	0.631	0.661	0.562
SE (RGB) [15]	0.60	0.61	0.56
SCG (RGB) [41]	0.55	0.57	0.46
HED-RGB [16]	0.701	0.716	0.704
HED-RGB-DA	**0.706**	**0.726**	**0.709**

Network architecture and training. Two different CNNs are considered in this experiment: a depth map prediction network and a contour detection network. The depth map prediction network follows the approach from [29] and it is trained on a subset of the train scenes of NYUv2, comprising 15k frames. Once trained, we use this network to generate depth maps for the contour detection network. The ground-truth depth information available from the dataset is *only* used to train the first network, while it is completely ignored when training and testing the second one. As a contour detection network we adopt the Holistically Nested Edge detection network of Xie *et al.* [16] in its RGB variant (HED-RGB). This is a VGG-based, fully-convolutional, deeply-supervised architecture with five "side-outputs" producing edge maps at different levels of granularity, which are then averaged at test time to compute the final prediction. In our DaConv version of this network (HED-RGB-DA) we include 3 DaConv blocks as described in Table I.

We train both HED-RGB and HED-RGB-DA using the following schedule: 20k iterations with learning rate 10^{-4} followed by 10k iterations with learning rate 10^{-5}, batch size equal to 20. As training data, we extract 16 randomly rotated and scaled patches of resolution 256×256 from each image in the annotated training set, replacing the original depth information with the one computed with the depth map prediction network.

Results. Table III compares the performance of HED-RGB with our DaConv version. As observed in previous application scenarios, even if the depth images are estimated and not provided as input, the proposed framework exhibits more accurate detections with respect to basic HED-RGB. However,

probably due to the noise introduced by the depth prediction phase, the performance gain is less prominent in this task. As a reference, we also report the performance of other learning-based methods that consider RGB-only edge detection on the NYUv2 database, *i.e.* the structured forest (SE) detector [15], the globalPb (gPb) [40] contour detector and the sparse code gradient (SCG) method [41]. As all these approaches are based on hand-crafted features, it is not surprising that CNN-based detectors such as HED outperform them by a large margin.

It is worth nothing that we trained HED-RGB and HED-RGB-DA using our implementation of the data augmentation scheme described in [16], as the publicly available code shared by the authors does not contain the data augmentation routines. Therefore, our results in Table III slightly differ from the ones reported in the original paper [16].

V. Conclusions

We presented DaConv, a novel general-purpose block for CNN architectures which learns scale-aware feature representations from RGB-D data. Our extensive experimental evaluation, conducted on publicly available datasets for three different robotics tasks (*i.e.* affordance detection, object coordinate regression and contour detection) demonstrated the flexibility and the effectiveness of the proposed block, which systematically boosts the performance of the network embedding it. We showed that the proposed DaConv block can be successfully employed even when the depth images are not available, but are "hallucinated" from the associated RGB inputs. Furthermore, by adopting a fully-convolutional network equipped with DaConv we outperformed state-of-the-art methods on the RGB-D Part Affordance Dataset [12]. In this work we considered pixel-level prediction tasks. However, we expect the proposed DaConv block to be effective also for other applications, *e.g.* involving image-level classification problems. Future works will be devoted to investigate this possibility.

References

[1] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *NIPS*, 2014.

[2] S. Yang, D. Maturana, and S. Scherer, "Real-time 3d scene layout from a single image using convolutional neural networks," in *ICRA*, 2016.

[3] P. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi, "Street-view change detection with deconvolutional networks," in *RSS*, 2016.

[4] N. Suenderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford, "Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free," in *RSS*, 2015.

[5] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," *ICRA*, 2016.

[6] A. Rubio, M. Villamizar, L. Ferraz, A. Penate-Sanchez, A. Ramisa, E. Simo-Serra, A. Sanfeliu, and F. Moreno-Noguer, "Efficient monocular pose estimation for complex 3d models," in *ICRA*, 2015.

[7] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *ECCV*, 2014.

[8] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *IROS*, 2015.

[9] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *IROS*, 2015.

[10] H. Schulz, N. Höft, and S. Behnke, "Depth and height aware semantic rgb-d perception with convolutional neural networks," in *ESANN*, 2015.

[11] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," *arXiv preprint arXiv:1608.02239*, 2016.

[12] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *ICRA*, 2015.

[13] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *ECCV*, 2014.

[14] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6d pose estimation in rgb-d images," in *ICCV*, 2015.

[15] P. Dollar and L. Zitnick, "Structured forests for fast edge detection," in *ICCV*, 2013.

[16] S. Xie and Z. Tu, "Holistically-nested edge detection," in *ICCV*, 2015.

[17] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *NIPS*, 2015.

[18] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE TPAMI*, vol. 35, no. 8, pp. 1872–1886, 2013.

[19] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," in *ICLR*, 2013.

[20] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," *CVPR*, 2016.

[21] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik, "Aligning 3d models to rgb-d images of cluttered scenes," in *CVPR*, 2015.

[22] A. Wang, J. Cai, J. Lu, and T.-J. Cham, "Mmss: Multi-modal sharable and specific feature learning for rgb-d object recognition," in *ICCV*, 2015.

[23] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *CVPR*, 2015.

[24] R. Gens and P. M. Domingos, "Deep symmetry networks," in *NIPS*, 2014.

[25] D. Laptev, N. Savinov, J. M. Buhmann, and M. Pollefeys, "Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks," *arXiv preprint arXiv:1604.06318*, 2016.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *ECCV*, 2014.

[27] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," *arXiv preprint arXiv:1511.03339*, 2015.

[28] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber, "Deep networks with internal selective attention through feedback connections," in *Advances in Neural Information Processing Systems*, 2014, pp. 3545–3553.

[29] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *ICCV*, 2015.

[30] S. Li, J. Xing, Z. Niu, S. Shan, and S. Yan, "Shape driven kernel adaptation in convolutional neural network for robust facial traits recognition," in *CVPR*, 2015.

[31] A. C. Muller and S. Behnke, "Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images," in *ICRA*, 2014.

[32] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets*. Springer, 1990, pp. 286–297.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ICML*, 2015.

[34] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *ICCV*, 2011.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[36] V. Badrinarayanan, A. Handa, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," *arXiv preprint arXiv:1505.07293*, 2015.

[37] J. Jose Tarrio and S. Pedre, "Realtime edge-based visual odometry for a monocular camera," in *CVPR*, 2015.

[38] L. Bose and A. Richards, "Fast depth edge detection and edge based rgb-d slam," in *ICRA*, 2016.

[39] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[40] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE TPAMI*, vol. 33, no. 5, pp. 898–916, 2011.

[41] R. Xiaofeng and L. Bo, "Discriminatively trained sparse code gradients for contour detection," in *NIPS*, 2012.